

[illegible]

```
0001 0 %TITLE 'DECnet Event Logger Main Module'
0002 0 MODULE EVLMAIN (
0003 0     LANGUAGE (BLISS32),
0004 0     IDENT = 'V04-000',
0005 0     MAIN = EVLSMAIN
0006 0 ) =
0007 1 BEGIN
0008 1
0009 1
0010 1 *****
0011 1 *
0012 1 *   COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0013 1 *   DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0014 1 *   ALL RIGHTS RESERVED.
0015 1 *
0016 1 *   THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0017 1 *   ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0018 1 *   INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0019 1 *   COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0020 1 *   OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0021 1 *   TRANSFERRED.
0022 1 *
0023 1 *   THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0024 1 *   AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0025 1 *   CORPORATION.
0026 1 *
0027 1 *   DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0028 1 *   SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0029 1 *
0030 1 *****
0031 1
0032 1
0033 1
0034 1 ++
0035 1 FACILITY:      DECnet Event Logging (EVL)
0036 1
0037 1 ABSTRACT:
0038 1
0039 1     This module contains the main entry for the event logger.
0040 1     A few routines of general utility are in the module too.
0041 1
0042 1 ENVIRONMENT:   VAX/VMS Operating System
0043 1
0044 1 AUTHOR:       Darrell Duffy   , CREATION DATE: 16-June-1980
0045 1
0046 1 MODIFIED BY:
0047 1
0048 1     V001      TMH0001      Tim Halvorsen   25-Jun-1981
0049 1     Purge working set to minimum before hibernating,
0050 1     to limit system physical memory usage while not
0051 1     doing anything.
0052 1     Change references to ASCID macro to use %ASCID.
0053 1 --
```



```
55 0054 1 %SBTTL 'Definitions'
56 0055 1
57 0056 1
58 0057 1 | TABLE OF CONTENTS:
59 0058 1 |
60 0059 1
61 0060 1 FORWARD ROUTINE
62 0061 1 EVLSMAIN : Main entry
63 0062 1 EVLSINITLOG : NOVALUE, : Initialize for debug logging
64 0063 1 EVLSPRINTLOG : NOVALUE : Print data messages to log file
65 0064 1 ;
66 0065 1
67 0066 1
68 0067 1 | INCLUDE FILES:
69 0068 1 |
70 0069 1
71 0070 1 LIBRARY 'LIB$:EVL$LIBRARY'; : EVL definitions
72 0071 1 LIBRARY 'SYSS$:LIBRARY:STARLET'; : VMS common definitions
73 0072 1
74 0073 1
75 0074 1 | OWN STORAGE:
76 0075 1 |
77 0076 1
78 0077 1 GLOBAL
79 0078 1 EVL$GL_LOGMASK : BBLOCK [8] : Logging control mask
80 0079 1 ;
81 0080 1
82 0081 1 |
83 0082 1 | EXTERNAL REFERENCES:
84 0083 1 |
85 0084 1
86 0085 1 EXTERNAL
87 0086 1 EVL$GW_NETSHOCHAN : WORD, : Channel to net for show
88 0087 1 EVL$B_RCVDONE : BYTE, : True if receive inactive
89 0088 1 EVL$B_TRANSDONE : BYTE; : True if transmitter inactive
90 0089 1
91 0090 1 EXTERNAL ROUTINE
92 0091 1 EVL$RECEIVE : NOVALUE, : Activate the receiver
93 0092 1 EVL$TRANSMIT : NOVALUE, : Activate the transmitter
94 0093 1 WKQ$DO_WORK_ITEM : : Perform a work item in queue
95 0094 1 EVL$OBTAINNETCHAN : NOVALUE : Obtain a channel to net
96 0095 1 ;
97 0096 1
98 0097 1 EXTERNAL ROUTINE
99 0098 1
100 0099 1 LIB$PUT_OUTPUT : ADDRESSING_MODE (GENERAL),
101 0100 1 LIB$CVT_HTB : ADDRESSING_MODE (GENERAL)
102 0101 1 ;
103 0102 1
```



```
105 0103 1 XSBTTL 'EVL$MAIN Main Entry'
106 0104 1 GLOBAL ROUTINE EVL$MAIN =
107 0105 1
108 0106 1 ++
109 0107 1 FUNCTIONAL DESCRIPTION:
110 0108 1
111 0109 1 This is the main entry point for the event logger.
112 0110 1 It calls the initialization routines
113 0111 1 and sits in a loop performing work from the work queue.
114 0112 1
115 0113 1 FORMAL PARAMETERS:
116 0114 1
117 0115 1 NONE
118 0116 1
119 0117 1 IMPLICIT INPUTS:
120 0118 1
121 0119 1 NONE
122 0120 1
123 0121 1 IMPLICIT OUTPUTS:
124 0122 1
125 0123 1 NONE
126 0124 1
127 0125 1 ROUTINE VALUE:
128 0126 1 COMPLETION CODES:
129 0127 1
130 0128 1 NONE
131 0129 1
132 0130 1 SIDE EFFECTS:
133 0131 1
134 0132 1 NONE
135 0133 1
136 0134 1 --
137 0135 1
138 0136 2 BEGIN
139 0137 2
140 0138 2 EVL$INITLOG (); ! Initialize for debug logging
141 0139 2 EVL$OBTAINNETCHAN (EVL$GW_NETSHOCHAN); ! Obtain a channel to net
142 0140 2 EVL$RECEIVE (); ! Initialize receiver portion
143 0141 2 EVL$TRANSMIT (); ! Initialize transmitter portion
144 0142 2
145 0143 3 WHILE NOT (.EVL$B_RCVDONE AND .EVL$GB_TRANSDONE) ! Until both sides finish,
146 0144 3 DO
147 0145 3 BEGIN
148 0146 3 $PURGWS(INADR=PLIT(0,%X'7FFFFFFF')); ! Purge entire working set
149 0147 3 $HIBER; ! Wait on some work to do
150 0148 3 WHILE WKQ$DO_WORK_ITEM () DO; ! Do work til empty queue
151 0149 3 END;
152 0150 2
153 0151 2 RETURN TRUE; ! Exit sucessfully
154 0152 1 END;
```

```
.TITLE EVLMAIN DECnet Event Logger Main Module
.IDENT \V04-000\
.PSECT $SPLITS,NOWRT,NOEXE,2
```

```
7FFFFFFF 00000002 00000 P.AAA: .LONG 2
00000000 00004 .LONG 0, 2147483647
.PSECT $GLOBALS,NOEXE,2

00000 EVL$GL_LOGMASK::
.BLKB 8

.EXTRN EVL$GW_NETSHOCHAN
.EXTRN EVL$B_RCVDONE, EVL$GB_TRANSDONE
.EXTRN EVL$RECEIVE, EVL$TRANSMIT
.EXTRN WKQ$DO_WORK_ITEM
.EXTRN EVL$OBTAINNETCHAN
.EXTRN LIB$PUT_OUTPUT, LIB$CVT_HTB
.EXTRN SYSS$PURGWS, SYSS$HIBER

.PSECT $CODE$,NOWRT,2

0000V CF 0000 00000 .ENTRY EVL$MAIN, Save nothing
0000G CF 0000G 00 FB 00002 CALLS #0, EVL$INITLOG
0000G CF 01 FB 00007 PUSHAB EVL$GW_NETSHOCHAN
0000G CF 00 FB 0000B CALLS #1, EVL$OBTAINNETCHAN
0000G CF 00 FB 00010 CALLS #0, EVL$RECEIVE
05 0000G CF E9 0001A 1$: CALLS #0, EVL$TRANSMIT
1C 0000G CF E8 0001F BLBC EVL$B_RCVDONE, 2$
00000000G 00 0000G CF 9F 00024 2$: BLBS EVL$GB_TRANSDONE, 4$
00000000G 00 01 FB 00028 PUSHAB P.AAA
0000G CF 00 FB 0002F CALLS #1, SYSS$PURGWS
DC 50 E9 0003B CALLS #0, SYSS$HIBER
50 F6 11 0003E CALLS #0, WKQ$DO_WORK_ITEM
01 D0 00040 3$: BLBC R0, 1$
04 00043 BRB 3$
MOVL #1, R0
RET
```

```
: 0104
: 0138
: 0139
: 0140
: 0141
: 0143
: 0146
: 0148
: 0151
: 0152
```

; Routine Size: 68 bytes, Routine Base: \$CODE\$ + 0000


```
156 0153 1 %SBTTL 'EVL$INITLOG      Initialization debug logging'
157 0154 1 ROUTINE EVL$INITLOG : NOVALUE =
158 0155 1
159 0156 1 ++
160 0157 1 FUNCTIONAL DESCRIPTION:
161 0158 1
162 0159 1     This routine initializes the internal logging flags for EVL debugging.
163 0160 1     The logical name EVL$LOG is translated to obtain a hex number which is
164 0161 1     converted to binary.  Each bit of the mask controls information to
165 0162 1     be logged.
166 0163 1
167 0164 1 FORMAL PARAMETERS:
168 0165 1
169 0166 1     NONE
170 0167 1
171 0168 1 IMPLICIT INPUTS:
172 0169 1
173 0170 1     EVL$LOG logical name
174 0171 1
175 0172 1 IMPLICIT OUTPUTS:
176 0173 1
177 0174 1     EVL$GL_LOGMASK
178 0175 1
179 0176 1 ROUTINE VALUE:
180 0177 1 COMPLETION CODES:
181 0178 1
182 0179 1     NONE
183 0180 1
184 0181 1 SIDE EFFECTS:
185 0182 1
186 0183 1     NONE
187 0184 1
188 0185 1 --
189 0186 1
190 0187 2 BEGIN
191 0188 2
192 0189 2 LITERAL
193 0190 2     RSLSIZE = 10                                ! Size of the result buffer
194 0191 2 ;
195 0192 2
196 0193 2 LOCAL
197 0194 2     RSLBFR : VECTOR [RSLSIZE, BYTE],          ! Buffer for the translation
198 0195 2     RSLDSC : VECTOR [2]                        ! Descriptor for the buffer
199 0196 2 ;
200 0197 2
201 0198 2 EVL$GL_LOGMASK = 0;                                ! Initialize the logging mask
202 0199 2 RSLDSC[0] = RSLSIZE;                            ! Setup the descriptor
203 0200 2 RSLDSC[1] = RSLBFR;
204 0201 2
205 0202 2 IF
206 0203 2     (
207 0204 2     $TRNLOG
208 0205 2     (
209 0206 2         LOGNAM = %ASCID 'EVL$LOG',
210 0207 2         RSLLEN = RSLDSC[0],
211 0208 2         RSLBUF = RSLDSC
212 0209 2     )
213 0210 2     )
214 0211 2
215 0212 2
```

```
213 0210 3 )
214 0211 3 EQL
215 0212 3 $$$_NORMAL
216 0213 3 THEN
217 0214 3 LIB$CVT_HTB
218 0215 3 (
219 0216 3 .RSLDSC [0],
220 0217 3 .RSLDSC [1],
221 0218 3 EVL$GL_LOGMASK
222 0219 3 )
223 0220 3 ;
224 0221 3
225 0222 3 END;
```

```
! If a successful translation
! Then convert the result
! Convert hex to binary

! Size of string
! Address of string
! Address of longword result
```

```
00 47 4F 4C 24 4C 56 45 0000C P.AAC: .PSECT $SPLITS,NOWRT,NOEXE,2
010E0007 00014 P.AAB: .ASCII \EVL$LOG\<0>
00000000' 00018 .LONG 17694727
. ADDRESS P.AAC
```

```
.EXTRN SYS$STRNLOG
```

```
.PSECT $CODE$,NOWRT,2
```

```
0000 00000 EVL$INITLOG:
5E 10 C2 00002 .WORD Save nothing 0154
0000' CF D4 00005 SUBL2 #16, SP
0A DD 00009 CLRL EVL$GL_LOGMASK 0198
04 AE 08 AE 9E 0000B PUSHL #10 0199
7E 7C 00010 MOVAB RSLBFR, RSLDSC+4 0200
7E D4 00012 CLRQ -(SP) 0209
0C AE 9F 00014 CLRL -(SP)
10 AE 9F 00017 PUSHAB RSLDSC
0000' CF 9F 0001A PUSHAB RSLDSC
06 FB 0001E PUSHAB P.AAB
50 D1 00025 CALLS #6, SYS$STRNLOG 0211
11 12 00028 CMPL R0, #1
0000' CF 9F 0002A BNEQ 1$
08 AE DD 0002E PUSHAB EVL$GL_LOGMASK 0215
08 AE DD 00031 PUSHL RSLDSC+4 0217
00000000G 00 03 FB 00034 PUSHL RSLDSC 0216
04 0003B 1$ CALLS #3, LIB$CVT_HTB
RET 0222
```

```
; Routine Size: 60 bytes, Routine Base: $CODE$ + 0044
```



```
227 0223 1 XSBTTL 'EVLSPRINTLOG Print a Data Message to the Log'
228 0224 1 GLOBAL ROUTINE EVLSPRINTLOG (LOGBIT, HEADDSC, EXTRADSC, DATADSC) : NOVALUE =
229 0225 1
230 0226 1 ++
231 0227 1 FUNCTIONAL DESCRIPTION:
232 0228 1
233 0229 1     Check the logging control mask and if the corresponding bit is set
234 0230 1     then print the special message to the log file. The message
235 0231 1     has a header and optionally some extra text which explains the
236 0232 1     logged message.
237 0233 1
238 0234 1 FORMAL PARAMETERS:
239 0235 1
240 0236 1     LOGBIT          Value of the logging bit to control the logging
241 0237 1                   as a bit number
242 0238 1     HEADDSC         Address of a descriptor of the header text
243 0239 1     EXTRADSC        Address of a descriptor of the extra text (optional)
244 0240 1     DATADSC         Address of a descriptor of the data to be converted
245 0241 1                   and printed
246 0242 1
247 0243 1 IMPLICIT INPUTS:
248 0244 1
249 0245 1     EVL$GL_LOGCONTROL
250 0246 1
251 0247 1 IMPLICIT OUTPUTS:
252 0248 1
253 0249 1     NONE
254 0250 1
255 0251 1 ROUTINE VALUE:
256 0252 1 COMPLETION CODES:
257 0253 1
258 0254 1     NONE
259 0255 1
260 0256 1 SIDE EFFECTS:
261 0257 1
262 0258 1     NONE
263 0259 1
264 0260 1 --
265 0261 1 BEGIN
266 0262 1
267 0263 1 MAP
268 0264 1
269 0265 1     HEADDSC          : REF BBLOCK,
270 0266 1     EXTRADSC         : REF BBLOCK,
271 0267 1     DATADSC          : REF BBLOCK
272 0268 1
273 0269 1
274 0270 1 LITERAL
275 0271 1     FAOSIZ = 256          ! The print buffer
276 0272 1
277 0273 1
278 0274 1 LOCAL
279 0275 1     FAOBUF : VECTOR [FAOSIZ, BYTE], ! Print buffer
280 0276 1     FAOLST : VECTOR [100],          ! List of args to $FAOL
281 0277 1     OUTDSC : VECTOR [2],            ! Descriptor of the output line
282 0278 1     BYTES,                          ! Counter for bytes written
283 0279 1     CTR : SIGNED,                  ! A random counter
```

```
284 0280 2 PTR, ! A random pointer
285 0281 2 ITR_CNT ! Temporary iteration count
286 0282 2 ;
287 0283 2
288 0284 2
289 0285 2
290 0286 2 See if this text should be logged, and if not then return
291 0287 2
292 0288 2 IF NOT .EVLSGL_LOGMASK [0, .LOGBIT, 1, 0]
293 0289 2 THEN
294 0290 2 RETURN
295 0291 2 ;
296 0292 2
297 0293 2 OUTDSC [0] = FAOSIZ; ! Initialize the output buffer dsc
298 0294 2 OUTDSC [1] = FAOBUF;
299 0295 2 FAOLST [0] = .HEADDSC; ! Header text
300 0296 2 FAOLST [1] = .DATADSC [DSC$W_LENGTH]; ! Data length
301 0297 2 FAOLST [2] = ; ! Extra text dsc
302 0298 2 (
303 0299 2 IF .EXTRADSC EQL 0
304 0300 2 THEN
305 0301 2 XASCID ''
306 0302 2 ELSE
307 0303 2 .EXTRADSC
308 0304 2 );
309 0305 2 $FAOL ! Write the header out
310 0306 2 (
311 0307 2 CTRSTR = XASCID '!' / !AS (length = !UL bytes)! / !AS! / ',
312 0308 2 OUTLEN = OUTDSC [0],
313 0309 2 OUTBUF = OUTDSC,
314 0310 2 PRMLST = FAOLST
315 0311 2 );
316 0312 2 LIB$PUT_OUTPUT (OUTDSC);
317 0313 2
318 0314 2 CTR = .DATADSC [DSC$W_LENGTH]; ! Size of message
319 0315 2 PTR = .DATADSC [DSC$A_POINTER]; ! Its address
320 0316 2 WHILE .CTR GTR 0 ! Process it all
321 0317 2 DO
322 0318 2 BEGIN
323 0319 2 OUTDSC [0] = FAOSIZ; ! Set a descriptor
324 0320 2 OUTDSC [1] = FAOBUF;
325 0321 2 ITR_CNT = MIN (.CTR, 20); ! Get byte count
326 0322 2 FAOLST [0] = .ITR_CNT; ! Add count parameter
327 0323 2 FAOLST [.ITR_CNT+1] = .ITR_CNT;
328 0324 2 FAOLST [(.ITR_CNT+1)*2] = .ITR_CNT;
329 0325 2 INCRU IDX FROM 1 TO .ITR_CNT ! A few bytes at a time
330 0326 2 DO
331 0327 2 BEGIN
332 0328 2 FAOLST [.IDX] = (.PTR) <0, 8, 0>; ! One for the hex
333 0329 2 FAOLST [.IDX + .ITR_CNT+1] = (.PTR) <0, 8, 0>; ! Decimal
334 0330 2 FAOLST [2*(.IDX + .ITR_CNT)+1] = 1; ! One for character
335 0331 2 FAOLST [2*(.IDX + .ITR_CNT)+1 + 1] = .PTR;
336 0332 2 PTR = .PTR + 1; ! Next one
337 0333 2 CTR = .CTR - 1; ! One less
338 0334 2 END
339 0335 2 ;
340 0336 2
```



```
... 341      P 0337      SFAOL      ! Saviour of bored programmers
    342      P 0338      (
    343      P 0339      CTRSTR = %ASCID '!(4XB)!/!(4UB)!/  !(4AF)!/' ,
    344      P 0340      OUTLEN = OUTDSC [0],
    345      P 0341      OUTBUF = OUTDSC,
    346      P 0342      PRMLST = FAOLST
    347      P 0343      );
    348      P 0344
    349      P 0345      LIB$PUT_OUTPUT (OUTDSC);      ! Write to SYSS$OUTPUT
    350      P 0346      END;
    351      P 0347      END;
```

```
74 67 6E 65 6C 28 20 20 53 41 21 20 20 2F 21 0001C P.AAE: .BLKB 0
21 29 73 65 74 79 62 20 4C 55 21 20 3D 20 68 0001C P.AAD: .LONG 17694720
                                00000000' 00020 P.AAG: .ADDRESS P.AAE
                                00024 P.AAG: .ASCII \!/ !AS (length = !UL bytes)!/ !AS!/-
                                00033 \<0>
                                00042
                                00048
                                010E0026 0004C P.AAF: .ASCII <0>
                                00000000' 00050 P.AAF: .LONG 17694758
                                00054 P.AAI: .ADDRESS P.AAG
                                00063 P.AAI: .ASCII \!(4XB)!/!(4UB)!/  !(4AF)!/\<0><0>
                                00072
                                00074 P.AAH: .LONG 17694750
                                00078 P.AAH: .ADDRESS P.AAI
```

.PSECT \$SPLITS,NOWRT,NOEXE,2

.EXTRN SYSS\$FAOL

.PSECT \$CODE\$,NOWRT,2

```
01 0000' 57 00000000G 00 00FC 00000 00000 .ENTRY EVL$PRINTLOG, Save R2,R3,R4,R5,R6,R7      : 0224
    56 00000000G 00 00 9E 00002 MOVAB LIB$PUT_OUTPUT, R7
    5E FD68 CE 9E 00009 MOVAB SYSS$FAOL, R6
    CF 04 AC E0 00010 MOVAB -664(SP), SP
    6E 0100 8F 3C 0001D 1$: BBS LOGBIT, EVL$GL_LOGMASK, 1$      : 0288
    04 AE FF00 CD 9E 00022 MOVZWL #256, OUTDSC
    08 AE 08 AC D0 00028 MOVAB FAOBUF, OUTDSC+4
    52 10 AC D0 0002D MOVL HEADDSC, FAOLST
    0C AE 62 3C 00031 MOVL DATADSC, R2
    0C AC D5 00035 MOVZWL (R2), FAOLST+4
    07 12 00038 TSTL EXTRADSC
    50 0000' CF 9E 0003A BNEQ 2$
    04 11 0003F MOVAB P.AAD, R0
    50 0C AC D0 00041 2$: BRB 3$
    10 AE 50 D0 00045 3$: MOVL EXTRADSC, R0
    08 AE 9F 00049 MOVL R0, FAOLST+8
    04 AE 9F 0004C PUSHAB FAOLST
    08 AE 9F 0004F PUSHAB OUTDSC
    0000' CF 9F 00052 PUSHAB OUTDSC
    66 04 FB 00056 CALLS #4, SYSS$FAOL      : 0311
```

			5E DD 00059	PUSHL SP		: 0312
	67		01 FB 0005B	CALLS #1, LIB\$PUT_OUTPUT		:
	53		62 3C 0005E	MOVZWL (R2), CTR		: 0314
	55	04	A2 D0 00061	MOVL 4(R2), PTR		: 0315
			53 D5 00065	TSTL CTR		: 0316
			72 15 00067	BLEQ 8\$		:
	6E	0100	8F 3C 00069	MOVZWL #256, OUTDSC		: 0319
04	AE	FF00	CD 9E 0006E	MOVAB FA0BUF, OUTDSC+4		: 0320
	50		53 D0 00074	MOVL CTR, R0		: 0321
	14		50 D1 00077	CMPL R0, #20		:
			03 15 0007A	BLEQ 5\$		:
	50		14 D0 0007C	MOVL #20, R0		:
	52		50 D0 0007F	MOVL R0, ITR_CNT		:
08	AE		52 D0 00082	MOVL ITR_CNT, FAOLST		: 0322
OC	AE42		52 D0 00086	MOVL ITR_CNT, FAOLST+4[ITR_CNT]		: 0323
50	52		01 78 0008B	ASHL #1, ITR_CNT, R0		: 0324
10	AE40		52 D0 0008F	MOVL ITR_CNT, FAOLST+8[R0]		:
	51		01 D0 00094	MOVL #1, -IDX		: 0325
			26 11 00097	BRB 7\$		:
08	AE41		65 9A 00099	MOVZBL (PTR), FAOLST[IDX]		: 0328
50	51		52 C1 0009E	ADDL3 ITR_CNT, IDX, R0		: 0329
OC	AE40		65 9A 000A2	MOVZBL (PTR), FAOLST+4[R0]		:
	54	8142	9E 000A7	MOVAB (IDX)+[ITR_CNT], R4		: 0330
50	54		01 78 000AB	ASHL #1, R4, R0		:
OC	AE40		01 D0 000AF	MOVL #1, FAOLST+4[R0]		:
50	54		01 78 000B4	ASHL #1, R4, R0		: 0331
	10	AE40	85 9E 000B8	MOVAB (PTR)+, FAOLST+8[R0]		:
	52		53 D7 000BD	DECL CTR		: 0333
			51 D1 000BF	CMPL IDX, ITR_CNT		: 0325
		08	D5 1B 000C2	BLEQU 6\$		:
		04	AE 9F 000C4	PUSHAB FAOLST		: 0343
		08	AE 9F 000C7	PUSHAB OUTDSC		:
		0000	AE 9F 000CA	PUSHAB OUTDSC		:
	66		CF 9F 000CD	PUSHAB P.AAH		:
			04 FB 000D1	CALLS #4, SYSSFAOL		:
	67		5E DD 000D4	PUSHL SP		: 0345
			01 FB 000D6	CALLS #1, LIB\$PUT_OUTPUT		:
			8A 11 000D9	BRB 4\$		: 0316
			04 000DB	RET		: 0347

; Routine Size: 220 bytes, Routine Base: \$CODE\$ + 0080

:

:

:

:

:

:

PSECT SUMMARY

Name	Bytes	Attributes
\$GLOBALS	8	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$SPLITS	124	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$CODES	348	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)

:

:

:

:

:

:

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
\$255\$DUA28:[EVL.OBJ]EVL\$LIBRARY.L32;1	191	2	1	14	00:00.1
\$255\$DUA28:[SYS\$LIB]STARLET.L32;1	9776	9	0	581	00:01.3

:

:

:

:

:

:

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:EVLMAIN/OBJ=OBJ\$:EVLMAIN MSRC\$:EVLMAIN/UPDATE=(ENH\$:EVLMAIN)

Size: 348 code + 132 data bytes
Run Time: 00:08.7
Elapsed Time: 00:20.2
Lines/CPU Min: 2398
Lexemes/CPU-Min: 18522
Memory Used: 86 pages
Compilation Complete

0156 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

